Data Gathering

Example of gathering image data using webcam

```
import cv2
key = cv2. waitKey(1)
webcam = cv2.VideoCapture(0)

while True:
  try:
    check, frame = webcam.read()
    print(check) #prints true as long as the webcam is running
    print(frame) #prints matrix values of each framecd
    cv2.imshow("Capturing", frame)
    key = cv2.waitKey(1)

    if key == ord('s'):
      cv2.imwrite(filename='saved_img.jpg', img=frame)
      webcam.release()
      img_new = cv2.imread('saved_img.jpg', cv2.IMREAD_GRAYSCALE)
      img_new = cv2.imshow("Captured Image", img_new)
      cv2.waitKey(1650)
      cv2.destroyAllWindows()
      print("Processing image...")
      img_ = cv2.imread('saved_img.jpg', cv2.IMREAD_ANYCOLOR)
      print("Converting RGB image to grayscale...")
      gray = cv2.cvtColor(img_, cv2.COLOR_BGR2GRAY)
      print("Converted RGB image to grayscale...")
      print("Resizing image to 28x28 scale...")
      img_ = cv2.resize(gray,(28,28))
      print("Resized...")
      img_resized = cv2.imwrite(filename='saved_img-final.jpg',
img=img_)
      print("Image saved!")

      break
    elif key == ord('q'):
      print("Turning off camera.")
      webcam.release()
      print("Camera off.")
      print("Program ended.")
      cv2.destroyAllWindows()
      break

  except(KeyboardInterrupt):
    print("Turning off camera.")
    webcam.release()
```

```
    print("Camera off.")
    print("Program ended.")
    cv2.destroyAllWindows()
    break
```

```
---------------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
Cell In[1], line 1
----> 1 import cv2
      2 key = cv2. waitKey(1)
      3 webcam = cv2.VideoCapture(0)

ModuleNotFoundError: No module named 'cv2'
```

Example of gathering voice data using microphone

```
!pip3 install sounddevice
```

```
Requirement already satisfied: sounddevice in c:\users\ke\anaconda3\
lib\site-packages (0.4.6)
Requirement already satisfied: CFFI>=1.0 in c:\users\ke\anaconda3\lib\
site-packages (from sounddevice) (1.16.0)
Requirement already satisfied: pycparser in c:\users\ke\anaconda3\lib\
site-packages (from CFFI>=1.0->sounddevice) (2.21)
```

```
!pip3 install wavio
```

```
Requirement already satisfied: wavio in c:\users\ke\anaconda3\lib\
site-packages (0.0.8)
Requirement already satisfied: numpy>=1.19.0 in c:\users\ke\anaconda3\
lib\site-packages (from wavio) (1.26.4)
```

```
!pip3 install scipy
```

```
Requirement already satisfied: scipy in c:\users\ke\anaconda3\lib\
site-packages (1.11.4)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in c:\users\ke\
anaconda3\lib\site-packages (from scipy) (1.26.4)
```

```
!apt-get install libportaudio2
```

```
# import required libraries
import sounddevice as sd
from scipy.io.wavfile import write
import wavio as wv

# Sampling frequency
freq = 44100
```

```
# Recording duration
duration = 5

# Start recorder with the given values
# of duration and sample frequency
recording = sd.rec(int(duration * freq),
 samplerate=freq, channels=2)

# Record audio for the given number of seconds
sd.wait()

# This will convert the NumPy array to an audio
# file with the given sampling frequency
write("recording0.wav", freq, recording)

# Convert the NumPy array to audio file
wv.write("recording1.wav", recording, freq, sampwidth=2)
```

Web Scraping

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. The web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.

Image Scraping using BeautifulSoup and Request

```
!pip install bs4
```

```
Requirement already satisfied: bs4 in c:\users\ke\anaconda3\lib\site-
packages (0.0.2)
Requirement already satisfied: beautifulsoup4 in c:\users\ke\
anaconda3\lib\site-packages (from bs4) (4.12.2)
Requirement already satisfied: soupsieve>1.2 in c:\users\ke\anaconda3\
lib\site-packages (from beautifulsoup4->bs4) (2.5)
```

```
pip install requests
```

```
Requirement already satisfied: requests in c:\users\ke\anaconda3\lib\
site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
ke\anaconda3\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ke\anaconda3\
lib\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ke\
anaconda3\lib\site-packages (from requests) (2.0.7)
```

```
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ke\
anaconda3\lib\site-packages (from requests) (2024.2.2)
Note: you may need to restart the kernel to use updated packages.
```

```
import requests
from bs4 import BeautifulSoup

def getdata(url):
  r = requests.get(url)
  return r.text

htmldata = getdata("https://www.google.com/")
soup = BeautifulSoup(htmldata, 'html.parser')
for item in soup.find_all('img'):
  print(item['src'])
```

```
/images/branding/googlelogo/1x/
googlelogo_white_background_color_272x92dp.png
```

```
pip install selenium
```

```
Requirement already satisfied: selenium in c:\users\ke\anaconda3\lib\
site-packages (4.18.1)
Requirement already satisfied: urllib3<3,>=1.26 in c:\users\ke\
anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium)
(2.0.7)
Requirement already satisfied: trio~=0.17 in c:\users\ke\anaconda3\
lib\site-packages (from selenium) (0.25.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\ke\
anaconda3\lib\site-packages (from selenium) (0.11.1)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\ke\
anaconda3\lib\site-packages (from selenium) (2024.2.2)
Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\
ke\anaconda3\lib\site-packages (from selenium) (4.9.0)
Requirement already satisfied: attrs>=23.2.0 in c:\users\ke\anaconda3\
lib\site-packages (from trio~=0.17->selenium) (23.2.0)
Requirement already satisfied: sortedcontainers in c:\users\ke\
anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\ke\anaconda3\lib\site-
packages (from trio~=0.17->selenium) (3.4)
Requirement already satisfied: outcome in c:\users\ke\anaconda3\lib\
site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\ke\
anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\users\ke\anaconda3\
lib\site-packages (from trio~=0.17->selenium) (1.16.0)
Requirement already satisfied: wsproto>=0.14 in c:\users\ke\anaconda3\
lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in c:\
users\ke\anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26-
```

```
>selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\ke\anaconda3\lib\
site-packages (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\ke\anaconda3\
lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium)
(0.14.0)
Note: you may need to restart the kernel to use updated packages.
```

Image Scraping using Selenium

```
!pip install selenium
!apt-get update # to update ubuntu to correctly run apt install
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin
import sys
sys.path.insert(0,'/usr/lib/chromium-browser/chromedriver')
from selenium import webdriver
import time
import requests
import shutil
import os
import getpass
import urllib.request
import io
import time
from PIL import Image
user = getpass.getuser()
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
driver =
webdriver.Chrome('chromedriver',chrome_options=chrome_options)
search_url = "https://www.google.com/search?q={q}&tbm=isch&tbs=sur
%3Afc&hl=en&ved=0CAIQpwVqFwoTCKCa1c6s4-
oCFQAAAAAdAAAAABAC&biw=1251&bih=568"
driver.get(search_url.format(q='Car'))

def scroll_to_end(driver):
  driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
  time.sleep(5)#sleep_between_interactions

def getImageUrls(name,totalImgs,driver):
  search_url = "https://www.google.com/search?q={q}&tbm=isch&tbs=sur
%3Afc&hl=en&ved=0CAIQpwVqFwoTCKCa1c6s4-
oCFQAAAAAdAAAAABAC&biw=1251&bih=568"
  driver.get(search_url.format(q=name))
  img_urls = set()
```

```
  img_count = 0
  results_start = 0

  while(img_count<totalImgs): #Extract actual images now
   scroll_to_end(driver)
   thumbnail_results =
driver.find_elements_by_xpath("//img[contains(@class,'Q4LuWd')]")
   totalResults=len(thumbnail_results)
   print(f"Found: {totalResults} search results. Extracting links
from{results_start}:{totalResults}")

  for img in thumbnail_results[results_start:totalResults]:
    img.click()
    time.sleep(2)
    actual_images = driver.find_elements_by_css_selector('img.n3VNCb')
  for actual_image in actual_images:
    if actual_image.get_attribute('src') and 'https' in
actual_image.get_attribute('src'):
      img_urls.add(actual_image.get_attribute('src'))
      img_count=len(img_urls)

  if img_count >= totalImgs:
    print(f"Found: {img_count} image links")
    break
  else:
    print("Found:", img_count, "looking for more image links ...")
    load_more_button = driver.find_element_by_css_selector(".mye4qd")

driver.execute_script("document.querySelector('.mye4qd').click();")
    results_start = len(thumbnail_results)
    return img_urls
def downloadImages(folder_path,file_name,url):
  try:
    image_content = requests.get(url).content
  except Exception as e:
    print(f"ERROR - COULD NOT DOWNLOAD {url} - {e}")
  try:
    image_file = io.BytesIO(image_content)
    image = Image.open(image_file).convert('RGB')
    file_path = os.path.join(folder_path, file_name)
    with open(file_path, 'wb') as f:
      image.save(f, "JPEG", quality=85)
    print(f"SAVED - {url} - AT: {file_path}")
  except Exception as e:
    print(f"ERROR - COULD NOT SAVE {url} - {e}")

def saveInDestFolder(searchNames,destDir,totalImgs,driver):
  for name in list(searchNames):
    path=os.path.join(destDir,name)
    if not os.path.isdir(path):
```

```
      os.mkdir(path)
    print('Current Path',path)
    totalLinks=getImageUrls(name,totalImgs,driver)
    print('totalLinks',totalLinks)
  if totalLinks is None:
    print('images not found for :',name)

  else:
    for i, link in enumerate(totalLinks):
      file_name = f"{i:150}.jpg"
      downloadImages(path,file_name,link)

searchNames=['cat']
destDir=f'/content/drive/My Drive/Colab Notebooks/Dataset/'
totalImgs=5
saveInDestFolder(searchNames,destDir,totalImgs,driver)
```

```
Requirement already satisfied: selenium in c:\users\ke\anaconda3\lib\
site-packages (4.18.1)
Requirement already satisfied: urllib3<3,>=1.26 in c:\users\ke\
anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium)
(2.0.7)
Requirement already satisfied: trio~=0.17 in c:\users\ke\anaconda3\
lib\site-packages (from selenium) (0.25.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\ke\
anaconda3\lib\site-packages (from selenium) (0.11.1)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\ke\
anaconda3\lib\site-packages (from selenium) (2024.2.2)
Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\
ke\anaconda3\lib\site-packages (from selenium) (4.9.0)
Requirement already satisfied: attrs>=23.2.0 in c:\users\ke\anaconda3\
lib\site-packages (from trio~=0.17->selenium) (23.2.0)
Requirement already satisfied: sortedcontainers in c:\users\ke\
anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\ke\anaconda3\lib\site-
packages (from trio~=0.17->selenium) (3.4)
Requirement already satisfied: outcome in c:\users\ke\anaconda3\lib\
site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\ke\
anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\users\ke\anaconda3\
lib\site-packages (from trio~=0.17->selenium) (1.16.0)
Requirement already satisfied: wsproto>=0.14 in c:\users\ke\anaconda3\
lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in c:\
users\ke\anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26-
>selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\ke\anaconda3\lib\
site-packages (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\ke\anaconda3\
```

```
lib\site-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium)
(0.14.0)

'apt-get' is not recognized as an internal or external command,
operable program or batch file.
'apt' is not recognized as an internal or external command,
operable program or batch file.
'cp' is not recognized as an internal or external command,
operable program or batch file.

---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[68], line 22
     20 chrome_options.add_argument('--no-sandbox')
     21 chrome_options.add_argument('--disable-dev-shm-usage')
---> 22 driver =
webdriver.Chrome('chromedriver',chrome_options=chrome_options)
     23 search_url = "https://www.google.com/search?
q={q}&tbm=isch&tbs=sur%3Afc&hl=en&ved=0CAIQpwVqFwoTCKCa1c6s4-
oCFQAAAAAdAAAAABAC&biw=1251&bih=568"
     24 driver.get(search_url.format(q='Car'))

TypeError: WebDriver.__init__() got an unexpected keyword argument
'chrome_options'
```

Web Scraping of Movies Information using BeautifulSoup

Identifying the URL structure In the image above, you can see that the URL has several parameters after the question mark: release_date

— Shows only the movies released in a specific year. sort

— Sorts the movies on the page. sort=num_votes,desc translates to sort by number of votes in a descending order. page

— Specifies the page number.ref_

— Takes us to the the next or the previous page. The reference is the page we are currently on. adv_nxt and adv_prv are two possible values. They translate to advance to the next page, and advance to the previous page, respectivel

```
from requests import get
url = 'https://www.imdb.com/search/title?
release_date=2017&sort=num_votes,desc&page=1'
useragent = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0
Safari/537.36"}
response = get(url, headers= useragent)
print(response.text[:500])
```

```
<!DOCTYPE html><html lang="en-US"
xmlns:og="http://opengraphprotocol.org/schema/"
xmlns:fb="http://www.facebook.com/2008/fbml"><head><meta charSet="utf-
8"/><meta name="viewport"
content="width=device-width"/><script>if(typeof uet === 'function')
{ uet('bb', 'LoadTitle', {wb: 1});
}</script><script>window.addEventListener('load', (event) => {
       if (typeof window.csa !== 'undefined' && typeof window.csa ===
'function') {
           var csaLatencyPlugin = window.csa('Content', {
```

Understanding the HTML structure of a single page

```
from bs4 import BeautifulSoup
html_soup = BeautifulSoup(response.text, 'html.parser')
headers = {'Accept-Language': 'en-US,en;q=0.8'}
type(html_soup)
```

```
bs4.BeautifulSoup
```

Using BeautifulSoup to parse the HTML content

To parse our HTML document and extract the 50 div containers, we'll use a Python module called BeautifulSoup, the most common web scraping module for Python . In the following code cell we wil l- : Import the BeautifulSoup class creator from the package b

s

- a. Parse response.text by creating a BeautifulSoup object, and assign this object to html_soup. The 'html.parser' argument indicates that we want to do the parsing u
- sing Python's built-in HTML parser.

Now let's use the find_all() method to extract all the div containers that have a class attribute of lister-item mode-advanced:

```
movie_containers = html_soup.find_all('div', class_ = 'sc-ab6fa25a-3
bVYfLY dli-parent')
print(type(movie_containers))
print(len(movie_containers))
```

```
<class 'bs4.element.ResultSet'>
50
```

We can access the first container, which contains information about a single movie, by using list notation on movie_containers.

```
first_movie = movie_containers[0]
first_movie
```

```
<div class="sc-ab6fa25a-3 bVYfLY dli-parent"><div class="sc-ab6fa25a-2
gOsifL"><div class="sc-e5a25b0f-0 jQjDIb dli-poster-container"><div
class="ipc-poster ipc-poster--base ipc-poster--dynamic-width ipc-sub-
grid-item ipc-sub-grid-item--span-2" role="group"><div aria-label="add
to watchlist" class="ipc-watchlist-ribbon ipc-focusable ipc-watchlist-
ribbon--s ipc-watchlist-ribbon--base ipc-watchlist-ribbon--loading
ipc-watchlist-ribbon--onImage ipc-poster__watchlist-ribbon"
role="button" tabindex="0"><svg class="ipc-watchlist-ribbon__bg"
height="34px" role="presentation" viewbox="0 0 24 34" width="24px"
xmlns="http://www.w3.org/2000/svg"><polygon class="ipc-watchlist-
ribbon__bg-ribbon" fill="#000000" points="24 0 0 0 0 32 12.2436611
26.2926049 24 31.7728343"></polygon><polygon class="ipc-watchlist-
ribbon__bg-hover" points="24 0 0 0 0 32 12.2436611 26.2926049 24
31.7728343"></polygon><polygon class="ipc-watchlist-ribbon__bg-shadow"
points="24 31.7728343 24 33.7728343 12.2436611 28.2926049 0 34 0 32
12.2436611 26.2926049"></polygon></svg><div class="ipc-watchlist-
ribbon__icon" role="presentation"><svg class="ipc-loader ipc-loader--
circle ipc-watchlist-ribbon__loader" data-testid="watchlist-ribbon-
loader" height="48px" role="presentation" version="1.1" viewbox="0 0
48 48" width="48px" xmlns="http://www.w3.org/2000/svg"><g class="ipc-
loader__container" fill="currentColor"><circle class="ipc-
loader__circle ipc-loader__circle--one" cx="24" cy="9"
r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--
two" cx="35" cy="14" r="4"></circle><circle class="ipc-loader__circle
ipc-loader__circle--three" cx="39" cy="24" r="4"></circle><circle
class="ipc-loader__circle ipc-loader__circle--four" cx="35" cy="34"
r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--
five" cx="24" cy="39" r="4"></circle><circle class="ipc-loader__circle
ipc-loader__circle--six" cx="13" cy="34" r="4"></circle><circle
class="ipc-loader__circle ipc-loader__circle--seven" cx="9" cy="24"
r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--
eight" cx="13" cy="14" r="4"></circle></g></svg></div></div><div
class="ipc-media ipc-media--poster-27x40 ipc-image-media-ratio--
poster-27x40 ipc-media--base ipc-media--poster-m ipc-poster__poster-
image ipc-media__img" style="width:100%"><img alt="Hugh Jackman in
Logan (2017)" class="ipc-image" loading="lazy" sizes="50vw, (min-
width: 480px) 34vw, (min-width: 600px) 26vw, (min-width: 1024px) 16vw,
(min-width: 1280px) 16vw"
src="https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhL
Tg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX140_CR0
,1,140,207_.jpg"
srcset="https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00Nj
dhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX140_
CR0,1,140,207_.jpg 140w,
https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NW
EtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX210_CR0,2,21
0,311_.jpg 210w,
```

https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX280_CR0,3,280,414_.jpg 280w" width="140"/> <a aria-label="View title page for Logan" class="ipc-lockup-overlay ipc-focusable" href="/title/tt3315342/?ref_=sr_i_1"> <h3 class="ipc-title__text">1. Logan</h3> 20172h 17mR-16 <svg class="ipc-icon ipc-icon--star-inline" fill="currentColor" height="24" role="presentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg"><path d="M12 20.1l5.82 3.682c1.066.675 2.37-.322 2.09-1.584l-1.543-6.926 5.146-4.667c.94-.85.435-2.465-.799-2.567l-6.773-.602L13.29.89a1.38 1.38 0 0 0-2.581 0l-2.65 6.53-6.774.602C.052 8.126-.453 9.74.486 10.59l5.147 4.666-1.542 6.926c-.28 1.262 1.023 2.26 2.09 1.585L12 20.099z"></path></svg>8.1 (<!-- -->827K<!-- -->)<button aria-label="Rate Logan" class="ipc-rate-button sc-e2dbc1a3-1 jboOQc ratingGroup--user-rating ipc-rate-button--unrated ipc-rate-button--base" data-testid="rate-button"><svg class="ipc-icon ipc-icon--star-border-inline" fill="currentColor" height="24" role="presentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg"><path d="M22.724 8.217l-6.786-.587-2.65-6.22c-.477-1.133-2.103-1.133-2.58 0l-2.65 6.234-6.772.573c-1.234.098-1.739 1.636-.8 2.446l5.146 4.446-1.542 6.598c-.28 1.202 1.023 2.153 2.09 1.51l5.818-3.495 5.819 3.509c1.065.643 2.37-.308 2.089-1.51l-1.542-6.612 5.145-4.446c.94-.81.45-2.348-.785-2.446zm-10.726 8.89l-5.272 3.174 1.402-5.983-4.655-4.026 6.141-.531 2.384-5.634 2.398 5.648 6.14.531-4.654 4.026 1.402 5.983-5.286-3.187z"></path></svg>Rate</button> 77Metascore <button aria-disabled="false" aria-label="See more information about Logan" class="ipc-icon-button dli-info-icon ipc-icon-button--base ipc-icon-button--onAccent2"

```
role="button" tabindex="0" title="See more information about
Logan"><svg class="ipc-icon ipc-icon--info" fill="currentColor"
height="24" role="presentation" viewbox="0 0 24 24" width="24"
xmlns="http://www.w3.org/2000/svg"><path d="M0 0h24v24H0V0z"
fill="none"></path><path d="M11 7h2v2h-2zm0 4h2v6h-2zm1-9C6.48 2 2
6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm0 18c-4.41 0-8-
3.59-8-8s3.59-8 8-8 8 3.59 8 8-3.59 8-8
8z"></path></svg></button></div></div><div class="sc-ab6fa25a-1
bBwFsP"><div class="ipc-html-content ipc-html-content--base sc-
ab6fa25a-0 bhexuD dli-plot-container" role="presentation"><div
class="ipc-html-content-inner-div">In a future where mutants are
nearly extinct, an elderly and weary Logan leads a quiet life. But
when Laura, a mutant child pursued by scientists, comes to him for
help, he must get her to safety.</div></div></div></div>
```

The name of the movie

```
first_movie.div

<div class="sc-ab6fa25a-2 gOsifL"><div class="sc-e5a25b0f-0 jQjDIb
dli-poster-container"><div class="ipc-poster ipc-poster--base ipc-
poster--dynamic-width ipc-sub-grid-item ipc-sub-grid-item--span-2"
role="group"><div aria-label="add to watchlist" class="ipc-watchlist-
ribbon ipc-focusable ipc-watchlist-ribbon--s ipc-watchlist-ribbon--
base ipc-watchlist-ribbon--loading ipc-watchlist-ribbon--onImage ipc-
poster__watchlist-ribbon" role="button" tabindex="0"><svg class="ipc-
watchlist-ribbon__bg" height="34px" role="presentation" viewbox="0 0
24 34" width="24px" xmlns="http://www.w3.org/2000/svg"><polygon
class="ipc-watchlist-ribbon__bg-ribbon" fill="#000000" points="24 0 0
0 0 32 12.2436611 26.2926049 24 31.7728343"></polygon><polygon
class="ipc-watchlist-ribbon__bg-hover" points="24 0 0 0 0 32
12.2436611 26.2926049 24 31.7728343"></polygon><polygon class="ipc-
watchlist-ribbon__bg-shadow" points="24 31.7728343 24 33.7728343
12.2436611 28.2926049 0 34 0 32 12.2436611
26.2926049"></polygon></svg><div class="ipc-watchlist-ribbon__icon"
role="presentation"><svg class="ipc-loader ipc-loader--circle ipc-
watchlist-ribbon__loader" data-testid="watchlist-ribbon-loader"
height="48px" role="presentation" version="1.1" viewbox="0 0 48 48"
width="48px" xmlns="http://www.w3.org/2000/svg"><g class="ipc-
loader__container" fill="currentColor"><circle class="ipc-
loader__circle ipc-loader__circle--one" cx="24" cy="9"
r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--
two" cx="35" cy="14" r="4"></circle><circle class="ipc-loader__circle
ipc-loader__circle--three" cx="39" cy="24" r="4"></circle><circle
class="ipc-loader__circle ipc-loader__circle--four" cx="35" cy="34"
r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--
five" cx="24" cy="39" r="4"></circle><circle class="ipc-loader__circle
ipc-loader__circle--six" cx="13" cy="34" r="4"></circle><circle
class="ipc-loader__circle ipc-loader__circle--seven" cx="9" cy="24"
```

r="4"></circle><circle class="ipc-loader__circle ipc-loader__circle--eight" cx="13" cy="14" r="4"></circle></g></svg> <img alt="Hugh Jackman in Logan (2017)" class="ipc-image" loading="lazy" sizes="50vw, (min-width: 480px) 34vw, (min-width: 600px) 26vw, (min-width: 1024px) 16vw, (min-width: 1280px) 16vw"
src="https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX140_CR0,1,140,207_.jpg"
srcset="https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX140_CR0,1,140,207_.jpg 140w,
https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX210_CR0,2,210,311_.jpg 210w,
https://m.media-amazon.com/images/M/MV5BYzc5MTU4N2EtYTkyMi00NjdhLTg3NWEtMTY4OTEyMzJhZTAzXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_QL75_UX280_CR0,3,280,414_.jpg 280w" width="140"/> <a aria-label="View title page for Logan" class="ipc-lockup-overlay ipc-focusable" href="/title/tt3315342/?ref_=sr_i_1"> <h3 class="ipc-title__text">1. Logan</h3> 20172h 17mR-16 <svg class="ipc-icon ipc-icon--star-inline" fill="currentColor" height="24" role="presentation" viewbox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg"><path d="M12 20.1l5.82 3.682c1.066.675 2.37-.322 2.09-1.584l-1.543-6.926 5.146-4.667c.94-.85.435-2.465-.799-2.567l-6.773-.602L13.29.89a1.38 1.38 0 0 0-2.581 0l-2.65 6.53-6.774.602C.052 8.126-.453 9.74.486 10.59l5.147 4.666-1.542 6.926c-.28 1.262 1.023 2.26 2.09 1.585L12 20.099z"></path></svg>8.1 (<!-- -->827K<!-- -->)<button aria-label="Rate Logan" class="ipc-rate-button sc-e2dbc1a3-1 jboOQc ratingGroup--user-rating ipc-rate-button--unrated ipc-rate-button--base" data-testid="rate-button"><svg class="ipc-icon ipc-

```
icon--star-border-inline" fill="currentColor" height="24"
role="presentation" viewbox="0 0 24 24" width="24"
xmlns="http://www.w3.org/2000/svg"><path d="M22.724 8.217l-6.786-.587-
2.65-6.22c-.477-1.133-2.103-1.133-2.58 0l-2.65 6.234-6.772.573c-
1.234.098-1.739 1.636-.8 2.446l5.146 4.446-1.542 6.598c-.28 1.202
1.023 2.153 2.09 1.51l5.818-3.495 5.819 3.509c1.065.643 2.37-.308
2.089-1.51l-1.542-6.612 5.145-4.446c.94-.81.45-2.348-.785-2.446zm-
10.726 8.89l-5.272 3.174 1.402-5.983-4.655-4.026 6.141-.531 2.384-
5.634 2.398 5.648 6.14.531-4.654 4.026 1.402
5.983-5.286-3.187z"></path></svg><span class="ipc-rating-star--
rate">Rate</span></span></button></div><span class="sc-b0691f29-11
TmkKM"><span class="sc-b0901df4-0 bcQdDJ metacritic-score-box"
style="background-color:#54A72A">77</span><span class="metacritic-
score-label">Metascore</span></span></span></div><div class="sc-
ab6fa25a-4 ggHbBR dli-post-element"><button aria-disabled="false"
aria-label="See more information about Logan" class="ipc-icon-button
dli-info-icon ipc-icon-button--base ipc-icon-button--onAccent2"
role="button" tabindex="0" title="See more information about
Logan"><svg class="ipc-icon ipc-icon--info" fill="currentColor"
height="24" role="presentation" viewbox="0 0 24 24" width="24"
xmlns="http://www.w3.org/2000/svg"><path d="M0 0h24v24H0V0z"
fill="none"></path><path d="M11 7h2v2h-2zm0 4h2v6h-2zm1-9C6.48 2 2
6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm0 18c-4.41 0-8-
3.59-8-8s3.59-8 8-8 8 3.59 8 8-3.59 8-8
8z"></path></svg></button></div></div>
```

```
first_movie.a
```

```
<a aria-label="View title page for Logan" class="ipc-lockup-overlay
ipc-focusable" href="/title/tt3315342/?ref_=sr_i_1"><div class="ipc-
lockup-overlay__screen"></div></a>
```

```
first_movie.h3
```

```
<h3 class="ipc-title__text">1. Logan</h3>
```

```
first_movie.h3.a
```

```
first_name = first_movie.find('h3',class_ = 'ipc-
title__text').text[3:]
first_name
```

```
'Logan'
```

The year of the movie's release

```
first_year = first_movie.find('span', class_ = 'sc-b0691f29-8 ilsLEX
dli-title-metadata-item').text[:]
first_year
```

```
'2017'
```

The IMDB rating

```
rating = first_movie.find('span', class_='ipc-rating-star ipc-rating-
star--base ipc-rating-star--imdb ratingGroup--imdb-rating').text[:3]
rating
```

```
'8.1'
```

The Metascore

```
first_mscore = first_movie.find('span', class_="sc-b0901df4-0 bcQdDJ
metacritic-score-box")
first_mscore = int(first_mscore.text)
print(first_mscore)
```

```
77
```

The number of votes

```
votes = first_movie.find('span', class_='ipc-rating-star--
voteCount').text[2:6]
votes
```

```
'827K'
```

The script

```
# Lists to store the scraped data in
names = []
years = []
imdb_ratings = []
metascores = []
votes = []
# Extract data from individual movie container
for container in movie_containers:
    # If the movie has Metascore, then extract:
    if container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ
metacritic-score-box') is not None:
        # The name
        name = container.find('h3',class_ = 'ipc-
title__text').text[3:]
        names.append(name)
        # The year
        year = container.find('span', class_ = 'sc-b0691f29-8 ilsLEX
dli-title-metadata-item').text
        years.append(year)
        # The IMDB rating
        imdb = container.find('span', class_='ipc-rating-star ipc-
rating-star--base ipc-rating-star--imdb ratingGroup--imdb-
```

```
rating').text[:3]
        imdb_ratings.append(imdb)
        # The Metascore
        m_score = container.find('span', class_ = 'sc-b0901df4-0
bcQdDJ metacritic-score-box').text
        metascores.append((m_score))
        # The number of votes
        vote = container.find('span', class_='ipc-rating-star--
voteCount').text[2:6]
        votes.append(vote)

import pandas as pd
test_df = pd.DataFrame({'movie': names,
                        'year': years,
                        'imdb': imdb_ratings,
                        'metascore': metascores,
                        'votes': votes
                       })
print(test_df.info())
test_df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41 entries, 0 to 40
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movie      41 non-null     object
 1   year       41 non-null     object
 2   imdb       41 non-null     object
 3   metascore  41 non-null     object
 4   votes      41 non-null     object
dtypes: object(5)
memory usage: 1.7+ KB
None
```

	movie	year	imdb	metascore	votes
0	Logan	2017	8.1	77	827K
1	Thor: Ragnarok	2017	7.9	74	813K
2	Guardians of the Galaxy Vol. 2	2017	7.6	67	756K
3	Dunkirk	2017	7.8	94	736K
4	Spider-Man: Homecoming	2017	7.4	73	716K
5	Wonder Woman	2017	7.3	76	698K
6	Get Out	2017	7.8	85	

```
691K
7        Star Wars: Episode VIII - The Last Jedi  2017  6.9         84
670K
8                                Blade Runner 2049  2017  8.0         81
658K
9                                     Baby Driver  2017  7.5         86
605K
10                                             It  2017  7.3         69
603K
11                                           Coco  2017  8.4         81
586K
12     Three Billboards Outside Ebbing, Missouri  2017  8.1         88
553K
13                             John Wick: Chapter 2  2017  7.4         75
509K
14                                  Justice League  2017  6.1         45
477K
15                               The Shape of Water  2017  7.3         87
446K
16                  Jumanji: Welcome to the Jungle  2017  6.9         58
436K
17                     Kingsman: The Golden Circle  2017  6.7         44
361K
18                               Kong: Skull Island  2017  6.7         62
345K
19   Pirates of the Caribbean: Salazar's Revenge  2017  6.5         39
344K
20                             Beauty and the Beast  2017  7.1         65
333K
21                                       Lady Bird  2017  7.4         93
326K
22                             Call Me by Your Name  2017  7.8         94
313K
23                             The Greatest Showman  2017  7.5         48
310K
24                                  Alien: Covenant  2017  6.4         65
302K
25                    Murder on the Orient Express  2017  6.5         52
295K
26                  War for the Planet of the Apes  2017  7.4         82
280K
27                                       Wind River  2017  7.7         73
279K
28                                  Fast & Furious 8  2017  6.6         56
253K
29                                             Life  2017  6.6         54
252K
30                                         Mother!  2017  6.6         76
249K
```

```
31                          The Hitman's Bodyguard  2017  6.9         47
246K
32                                       I, Tonya  2017  7.5         77
242K
33               King Arthur: Legend of the Sword  2017  6.7         41
232K
34                             Ghost in the Shell  2017  6.3         52
227K
35                                  Darkest Hour  2017  7.4         75
220K
36                                 American Made  2017  7.1         65
207K
37                                 Atomic Blonde  2017  6.7         63
206K
38                                     The Mummy  2017  5.4         34
206K
39                                      Baywatch  2017  5.5         37
201K
40                                        Bright  2017  6.3         29
201K
```

The script for multiple pages

```
from time import time
from time import sleep
from random import randint
from IPython.core.display import clear_output
from requests import get
pages = [ '1','2','3','4','5']
years_url = [ '2017', '2018', '2019', '2020']
# Redeclaring the lists to store data in
names = []
years = []
imdb_ratings = []
metascores = []
votes = []
# Preparing the monitoring of the loop
start_time = time()
requests = 0
# For every year in the interval 2000-2017
for year_url in years_url:
    # For every page in the interval 1-4
    for page in pages:
        # Make a get request
        url = 'https://www.imdb.com/search/title?
release_date=2017&sort=num_votes,desc&page=1'
        useragent = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0
Safari/537.36"}
```

```
        response = get(url, headers= useragent)
        print(response.text[:500])
        # Pause the loop
        sleep(randint(8,15))
        # Monitor the requests
        requests += 1
        elapsed_time = time() - start_time
        print('Request:{}; Frequency: {} requests/s'.format(requests,
requests/elapsed_time))
        clear_output(wait = True)
        # Throw a warning for non-200 status codes
        if response.status_code != 200:
            print('Request: {}; Status code: {}'.format(requests,
response.status_code))
            # Break the loop if the number of requests is greater than
expected
        if requests > 72:
            print('Number of requests was greater than expected.')
            break
        # Parse the content of the request with BeautifulSoup
        page_html = BeautifulSoup(response.text, 'html.parser')
        # Select all the 50 movie containers from a single page
        mv_containers = page_html.find_all('div', class_ = 'sc-
ab6fa25a-3 bVYfLY dli-parent')
        # For every movie of these 50
        for container in mv_containers:
        # If the movie has a Metascore, then:
            if container.find('span', class_ = 'sc-b0901df4-0 bcQdDJ
metacritic-score-box') is not None:
                name = container.find('h3',class_ = 'ipc-
title__text').text[3:]
                names.append(name)
                year = container.find('span', class_ = 'sc-b0691f29-8
ilsLEX dli-title-metadata-item').text
                years.append(year)
                imdb = container.find('span', class_='ipc-rating-star
ipc-rating-star--base ipc-rating-star--imdb ratingGroup--imdb-
rating').text[:3]
                imdb_ratings.append(imdb)
                m_score = container.find('span', class_ = 'sc-
b0901df4-0 bcQdDJ metacritic-score-box').text
                metascores.append((m_score))
                vote = container.find('span', class_='ipc-rating-
star--voteCount').text[2:6]
                votes.append(vote)

<!DOCTYPE html><html lang="en-US"
xmlns:og="http://opengraphprotocol.org/schema/"
xmlns:fb="http://www.facebook.com/2008/fbml"><head><meta charSet="utf-
8"/><meta name="viewport"
```

```
content="width=device-width"/><script>if(typeof uet === 'function')
{ uet('bb', 'LoadTitle', {wb: 1});
}</script><script>window.addEventListener('load', (event) => {
        if (typeof window.csa !== 'undefined' && typeof window.csa ===
'function') {
            var csaLatencyPlugin = window.csa('Content', {

Request:20; Frequency: 0.06046874871419513 requests/s

movie_ratings = pd.DataFrame({'movie': names,
                              'year': years,
                              'imdb': imdb_ratings,
                              'metascore': metascores,
                              'votes': votes
                             })
print(movie_ratings.info())
movie_ratings.head(10)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 820 entries, 0 to 819
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movie      820 non-null    object
 1   year       820 non-null    object
 2   imdb       820 non-null    object
 3   metascore  820 non-null    object
 4   votes      820 non-null    object
dtypes: object(5)
memory usage: 32.2+ KB
None

                                     movie  year imdb metascore votes
0                                    Logan  2017  8.1        77  827K
1                            Thor: Ragnarok  2017  7.9        74  813K
2           Guardians of the Galaxy Vol. 2  2017  7.6        67  756K
3                                  Dunkirk  2017  7.8        94  736K
4                   Spider-Man: Homecoming  2017  7.4        73  716K
5                             Wonder Woman  2017  7.3        76  698K
6                                  Get Out  2017  7.8        85  691K
7  Star Wars: Episode VIII - The Last Jedi  2017  6.9        84  670K
8                         Blade Runner 2049  2017  8.0        81  658K
9                               Baby Driver  2017  7.5        86  605K

movie_ratings.tail(10)

                                  movie  year imdb metascore votes
810            The Hitman's Bodyguard  2017  6.9        47  246K
811                             I, Tonya  2017  7.5        77  242K
812   King Arthur: Legend of the Sword  2017  6.7        41  232K
```

```
813                  Ghost in the Shell  2017  6.3        52  227K
814                       Darkest Hour  2017  7.4        75  220K
815                      American Made  2017  7.1        65  207K
816                      Atomic Blonde  2017  6.7        63  206K
817                          The Mummy  2017  5.4        34  206K
818                           Baywatch  2017  5.5        37  201K
819                             Bright  2017  6.3        29  201K
```

```
movie_ratings.to_csv('movie_ratings.csv')
```

Data Preparation

Example of Data Preparation of movie_rating.csv

```
movie_ratings['year'].unique()
```

```
array(['2017'], dtype=object)
```

```
movie_ratings.dtypes
```

```
movie        object
year         object
imdb         object
metascore    object
votes        object
dtype: object
```

```
movie_ratings['year'] = (movie_ratings.year.apply(lambda
x:x.replace('(I)','')))
```

```
movie_ratings['year'] = (movie_ratings.year.apply(lambda
x:x.replace('(III)','')))
```

```
movie_ratings['year'].unique()
```

```
array(['2017'], dtype=object)
```

```
movie_ratings['year'] = (movie_ratings.year.apply(lambda
x:x.replace('(','')))
```

```
movie_ratings['year'].unique()
```

```
array(['2017'], dtype=object)
```

```
movie_ratings['year'] = (movie_ratings.year.apply(lambda
x:x.replace(')','')))
```

```
movie_ratings['year'].unique()
```

```
array(['2017'], dtype=object)
```

```
movie_ratings['year'] = movie_ratings['year'].astype(int)

movie_ratings['year'].unique()
```

```
array([2017])
```

```
movie_ratings.dtypes
```

```
movie        object
year          int32
imdb         object
metascore    object
votes        object
dtype: object
```

```
movie_ratings.head(10)
```

	movie	year	imdb	metascore	votes
0	Logan	2017	8.1	77	827K
1	Thor: Ragnarok	2017	7.9	74	813K
2	Guardians of the Galaxy Vol. 2	2017	7.6	67	756K
3	Dunkirk	2017	7.8	94	736K
4	Spider-Man: Homecoming	2017	7.4	73	716K
5	Wonder Woman	2017	7.3	76	698K
6	Get Out	2017	7.8	85	691K
7	Star Wars: Episode VIII - The Last Jedi	2017	6.9	84	670K
8	Blade Runner 2049	2017	8.0	81	658K
9	Baby Driver	2017	7.5	86	605K

```
movie_ratings
```

	movie	year	imdb	metascore	votes
0	Logan	2017	8.1	77	827K
1	Thor: Ragnarok	2017	7.9	74	813K
2	Guardians of the Galaxy Vol. 2	2017	7.6	67	756K
3	Dunkirk	2017	7.8	94	736K
4	Spider-Man: Homecoming	2017	7.4	73	716K
..	...	...	...	...	...
815	American Made	2017	7.1	65	207K
816	Atomic Blonde	2017	6.7	63	206K
817	The Mummy	2017	5.4	34	206K
818	Baywatch	2017	5.5	37	201K
819	Bright	2017	6.3	29	201K

```
[820 rows x 5 columns]
```